Firebase: trabajar en la nube

Jesús Tomás
Vicente Carbonell
Jordi Bataller
Jaime Lloret

Firebase Authentication, Cloud Firestore, Cloud Functions, Cloud Storage, Analytics, Crashlytics, Test Lab, Redes sociales y mucho más.

Marcombo

Alfaomega

Firebase: trabajar en la nube

Firebase: trabajar en la nube

Jesús Tomás
Vicente Carbonell
Jordi Bataller
Jaime Lloret

Marcombo

Alfaomega

ALFAOMEGA

Empresas del Grupo

Colombia: Alfaomega Colombiana S.A.
Calle 62 No.20-46 esquina, Bogotá
Teléfono (57-1) 746 0102 Fax: (57-1) 210 0122
cliente@alfaomegacolombiana.com

México: Alfaomega Grupo Editor S.A. de C.V.
Calle Doctor Olvera No. 74, Colonia Doctores,
Delegación Cuaúhtemoc, Ciudad de México
C.P. 06720 • teléfono (52-55) 5089 7740
Fax (52-55) 5575 2420
Sin costo 01-800-020-4396
libreriapitagoras@alfaomega.com.mx

Argentina: Alfaomega Grupo Editor Argentino S.A.
Av. Córdoba 1215, Piso 10
Capital Federal, Buenos Aires
Teléfono/Fax: (54-11) 4811 7183 / 8352 / 0887
ventas@alfaomegaeditor.com.ar

Chile: Alfaomega Grupo Editor S.A.
Av. Providencia 1443. Oficina 24, Santiago
Teléfonos (56-2) 2235 4248 / 2947 9351 / 2235 5786
agechile@alfaomega.cl

www.alfaomega.com.co

Firebase: trabajar en la nube
Bogotá, 2019

Diseño de la cubierta: Enedenú Diseño Gráfico

ISBN: 978-958-778-497-8 (Edición Colombia)
ISBN: 978-84-267-2660-5 (Edición España)

Hecho en Colombia
Printed and made in Colombia

Para Erica, con el amor de su padre

JESÚS TOMÁS

Para Maite, Inés y Carles, con todo el cariño, por su paciencia y comprensión

VICENTE CARBONELL

Para Mar, Ferran y Cristina

JORDI BATALLER

Para Cristina y Claudia, por el tiempo que no he pasado junto a vosotras

JAIME LLORET

Índice general

¿Cómo leer este libro?

Este libro se ha estructurado en varios capítulos que abordan principalmente el uso de Firebase para el desarrollo de aplicaciones en Android. No es precisa una lectura secuencial, el lector puede ir directamente al capítulo que le interese.

En este libro se da por supuesto que el lector tiene experiencia en programación sobre Android. No se tratan temas relativos al desarrollo básico como los componentes o estructura de las aplicaciones. Si el lector está interesado en un texto que aborde la programación en Android desde el principio, le recomendamos *El gran libro de Android* publicado en esta misma editorial. También puede ser de interés la lectura de otros libros de esta colección como *El gran libro de Android Avanzado*, *Plataformas Android: Wear, TV, Auto y Google Play Games* o *Android Things y Visión Artificial*.

El libro que tienes entre las manos no ha sido concebido solo para ser leído. Es más bien una guía estructurada que te irá proponiendo una serie de ejercicios, actividades, vídeos explicativos, test de autoevaluación, etc. Parte de este material y algunos recursos adicionales están disponibles en la web www.androidcurso.com. En ella se publicarán las novedades, erratas e información complementaria relativas a este libro. Además encontrarás:

- **Material adicional sobre Android:** Encontrarás nuevos tutoriales, vídeos, referencias, etc., no incluidos en el libro.
- **Código abierto de proyectos Android:** Muchos alumnos que han realizado cursos con nosotros han tenido la generosidad de compartir sus proyectos. Te recomendamos que consultes la lista de proyectos disponibles de código abierto: puedes aprender mucho estudiando su código.
- **Cursos online:** Si te interesa ampliar tu formación, puedes matricularte en cursos sobre Android impartidos por la Universidad Politécnica de Valencia. Incluso puedes obtener un título de Especialización o de Máster de forma 100 % online.

A lo largo del libro se utilizan los siguientes iconos para indicar distintos tipos de recursos:

Objetivos: Antes de empezar cada capítulo lee con detenimiento la introducción y los objetivos.

Ejercicio: La mejor forma de aprender es hacer ejercicios. No tendrás más que ir siguiendo los pasos uno tras otro para descubrir cómo se resuelve el ejercicio propuesto. Para que no se te haga pesado teclear todo el código, te sugerimos que lo copies y pegues desde la página web del curso.

Práctica: Este será el momento de que tomes la iniciativa y trates de resolver el problema que se propone. Recuerda que para aprender, hay que practicar.

Solución: Te será de ayuda si tienes dificultades al resolver una práctica o simplemente quieres comparar tu solución con otra diferente.

Vídeo [Tutorial]: Vídeos grabados por los autores donde se exponen de forma didáctica los aspectos clave de los temas tratados. Se utiliza una moderna herramienta desarrollada en la Universidad Politécnica de Valencia que te permitirá ver simultáneamente las presentaciones y al profesor.

Enlaces de interés: Internet te será de gran ayuda para completar la información necesaria para programar en Android. Te proponemos las páginas más interesantes de cada apartado.

Preguntas de repaso: ¿Has comprendido correctamente los aspectos clave? Sal de dudas haciendo los test de autoevaluación.

Trivial programación Android: Instálate esta app y mide tus conocimientos jugando en red contra otros oponentes.

CAPÍTULO 1.

Introducción a Firebase y Autentificación

JESÚS TOMÁS

Firebase es la nueva plataforma de Google con la que conseguiremos que nuestras aplicaciones trabajen en la nube. Toda la problemática que supone trabajar en Internet (almacenamiento, registros de usuarios, gestión del backend…) la tenemos resuelta de una forma sencilla e integrada. Ahorraremos cientos de horas de implementación y, además, al estar basado en la infraestructura de Google, obtendremos resultados altamente fiables, seguros y escalables. Entre los servicios que incorpora podemos destacar:

- Identificación de usuario
- Bases de datos NoSQL y de tiempo real
- Mensajería en la nube
- Notificaciones push
- Almacenamiento de ficheros y hosting
- Análisis en tiempo real de lo que hacen los usuarios
- Herramientas de monetización
- Y mucho más

Se trata de una plataforma disponible para diferentes sistemas (Android, iOS, Web, C++ e Unity). Además, el coste del servicio es gratuito para un uso moderado; solo tendremos que pagar cuando nuestra aplicación comience a tener éxito.

Google quiere centralizar en Firebase todas plataformas para la gestión de apps (Analytics, Admod, Parse…). Desde una misma consola tendremos acceso a todos los datos de nuestra aplicación.

Firebase puede englobarse como un servicio del tipo *Mobile Backend as a Service* (MBaaS). Empezaremos este capítulo describiendo algunas alternativas que ofrecen servicios similares.

Continuaremos con una introducción a Firebase, resaltaremos sus ventajas e inconvenientes. Pasaremos a realizar una descripción de cada uno de los servicios que integra la plataforma.

En la última parte del capítulo, aprenderemos a utilizar los servicios de autentificación de Firebase. Estos incluyen la posibilidad de pedir al usuario unas credenciales basadas en correo y contraseña o utilizar las credenciales que ya disponen en redes sociales.

Objetivos

- Comparar Firebase con otros servicios del tipo MBaaS.
- Enumerar los servicios integrados en Firebase.
- Describir el proceso de autentificación que proporciona Firebase.
- Implementar un proceso de autentificación automático con FirebaseUI.
- Implementar procesos de autentificación personalizados basados en correo y contraseña, Google, Facebook, Twitter y anónima.
- Describir en qué consiste la unificación de cuentas.

1.1. *Mobile Backend as a Service* (MBaaS)

Firebase puede ser catalogado como una plataforma de tipo *Mobile Backend as a Service* (MBaaS), por lo que puede ser interesante describir primero este tipo de plataformas y qué alternativas existen en la actualidad.

Un MBaaS proporciona una serie de servicios en la nube (autentificación, almacenamiento, notificaciones, análisis…) para el desarrollo de aplicaciones móviles y Web. La principal finalidad es aumentar la productividad en el desarrollo de apps, dado que el programador no ha de desarrollar los diferentes servicios en la nube que requiere su aplicación. Además, la plataforma nos proporciona todas las infraestructuras de acceso, almacenamiento y seguridad. Otra ventaja es que podemos gestionar todas nuestras aplicaciones desde una consola Web unificada (el Backend).

Aunque la mayoría de los MBaaS se enfocan al desarrollo de aplicaciones móviles, realmente, estos servicios también pueden ser usados para el desarrollo de aplicaciones Web o de escritorio. Por esta razón, también se conocen simplemente como *Backend as a Service* (BaaS), sin el término *Mobile* en el nombre.

Los MBaaS suelen estar formados por un SDK para diferentes plataformas o lenguajes y un Backend (normalmente Web) para la gestión de proyectos. El acceso a la plataforma desde los diferentes terminales suele realizarse mediante servicios Web tipo REST.

En la actualidad existen muchos MBaaS. De hecho, las principales empresas tecnológicas como Google, Apple, Amazon o Microsoft han sacado recientemente una plataforma con este propósito. Veamos algunos de ellos:

AWS Mobile (Amazon)

Amazon es una de las empresas pioneras en ofrecer servicios Web. Recientemente ha sacado una plataforma específica para apps: AWS Mobile. Entre sus servicios destacamos:

- DynamoDB: bases de datos no SQL
- Cognito: Autentificación de usuarios
- Notificaciones push
- Almacenamiento
- Lambda: funciones ejecutadas en servidor
- Kinesis: procesamiento de datos en tiempo real

Azure (Microsoft)

Dentro de los servicios de computación en la nube ofrecidos por Azure, se ha desarrollado una línea específica para dar servicio a aplicaciones móviles (*Mobile Apps*).

- Dispone de SDK para iOS, Android, Windows o Mac.
- Ofrece servicio de autentificación basado en cuentas de Microsoft, Google o las principales redes sociales.
- Permite enviar notificaciones push a cualquier plataforma.
- Se puede programar la lógica de la aplicación en el backend utilizando C# o Node*.js*.

CloudKit (Apple)

Esta plataforma de Apple para potenciar los servicios de iCloud en iOS y macOS.

Ofrece servicios de autentificación, una base de datos privada, una base de datos pública y servicios de almacenamiento de datos. Para el desarrollo en Android no sería una alternativa viable.

Parse (Facebook)

Esta plataforma fue potenciada inicialmente por Facebook, pero en la actualidad han abandonado el proyecto y han pasado el testigo a una comunidad de código abierto. De hecho, una de las principales ventajas de esta plataforma es su versión en código abierto, tanto para el servidor como para SDK de decenas de plataformas (http://parseplatform.org/). En este caso ha de ser el desarrollador quien proporcione el servidor, pero, claro, conseguimos un servicio ilimitado sin coste por uso. Integra: MongoDB como base de datos, Amazon S3 bucket para almacenar ficheros y OneSignal para las notificaciones.

La siguiente tabla muestra una comparativa de estas plataformas:

	Firebase	AWS	CloudKit	Azure	Parse
Empresa	Google	Amazon	Apple	Microsoft	Facebook
Código abierto	no	no	no	no	sí
API nativo	Android iOS Unity JavaScript REST C	REST (Java JavaScript Perl PHP Python Ruby)	iOS Mac	Android iOS Windows Mac	Android iOS Unity JavaScript REST C Arduino *.net* PHP
Bases de datos	NoSQL	NoSQL /SQL	¿?	NoSQL /SQL	¿?

La siguiente tabla muestra una comparativa de los servicios ofrecidos sin coste:

	Firebase	AWS*	Azure**	Parse
Autentificación	-	-	500 K	-
Notificaciones	-	1 M /m 5 K usuarios/m	1 M	-
Cloud Messaging	-	1M mensajes/m	-	-
Almacenamiento	5 GB 1 GB/día	10 GB	1 GB	-
Hosting	1 GB 10 GB/m (dominio, SSL)	10 GB	1 GB (PHP)	-
Base de datos	1 GB 10 GB/m 100 conexiones	25 GB 200M solicit/m (SQL, NoSQL)	20 MB* (SQL, NoSQL)	-
Funciones	125K solicit/m 40K GB-seg/m	1M solicit/m 400K GB-seg/m	1M solicit 400K GB-seg	-
Test dispositivo	5 test/d real 10 test/d virtual	250 min	no	no

(-) servicio ilimitado

(*) El servicio gratuito solo se ofrece durante el primer año.
Amazon limita el uso de máquinas virtuales a 750 horas el primer año.

(**) Un máximo de 60 minutos de CPU/día y 1 GB de RAM.

Enlaces de interés

- http://searchmicroservices.techtarget.com/tip/Firebase-AWS-and-more-A-comparison-of-five-leading-MBaaS-providers

- https://www.infoworld.com/article/2842791/application-development/mbaas-shoot-out-5-cloud-platforms-for-building-mobile-apps.html
- https://db-engines.com/en/system/Amazon+DynamoDB%3BFirebase+Realtime+Database

1.2. Introducción a Firebase

Firebase empieza como una empresa, fundada en el año 2011, en San Francisco. Su finalidad era dar servicios de backend en el desarrollo de aplicaciones. Su producto inicial fue la base de datos en tiempo real, aunque luego se añadieron nuevos servicios. En el año 2014 fue comprada por Google y fue lanzada en el Google I/O del 2016, donde se integraron nuevos servicios con otros que antes ofrecía Google de forma independiente (Analytics, AdMob…).

Trabajar con Firebase nos aporta gran cantidad de ventajas:

- Una sola herramienta nos resuelve los principales problemas de trabajo en la nube: bases de datos, autentificación, almacenamiento, análisis detallados de nuestra app, etc.
- Sencillo y rápido de utilizar.
- Podemos utilizarlo desde múltiples plataformas. Tiene soporte para Android, iOS, Web (JavaScript), C++ e Unity, aunque podemos usarla desde cualquier sistema gracias a su API REST.
- Gratuito para un volumen de uso moderado.
- Firebase utiliza la infraestructura de Google, una de las más extensas y fiables de la actualidad. La responsabilidad de la seguridad y fiabilidad, recae ahora en Google.
- Escalable Google dispone de una extensa red de servicios en la nube.

También podemos destacar varias desventajas:

- La versión gratuita permite acceder a todas las funciones de Firebase, pero con algunas limitaciones. Por ejemplo, estamos limitados a 100 conexiones simultáneas en las bases de datos y a la autentificación de 10 K usuarios al mes. Estas limitaciones nos permiten trabajar con muchos tipos de aplicaciones. Como alternativas al plan gratuito (Spark) disponemos de dos planes: Flame (25 $/mes) y Blaze (pago por uso)[1].
- Al tratarse de un producto comercial (no es de código abierto) estamos expuestos a posibles cambios en las condiciones de comercialización.
- Las bases de datos NoSQL tienen sus ventajas, pero también presentan inconvenientes. El principal es que nos obliga a cambiar la forma de

[1] https://firebase.google.com/pricing/?hl=es-419

concebir las estructuras de datos. Además, presenta limitaciones a la hora de realizar búsquedas.

1.2.1. Los servicios de Firebase

Firebase integra diferentes servicios en una misma plataforma. En este apartado describiremos brevemente cada uno de ellos:

1.2.1.1. Herramientas de desarrollo

Authentication: Resuelve el problema de la autentificación de tus usuarios de una forma sencilla y sin riesgo de que su seguridad quede comprometida. Google se encarga de todo.

Puedes usar una autentificación basada en correo y contraseña o a través del número de teléfono. También puedes usar las principales redes sociales, incluidas Google, Facebook y Twitter y GitHub. Puedes gestionar tus usuarios de forma muy sencilla desde el backend.

Si utilizas una interfaz de usuario prediseñado (Firebase UI), puedes resolver la autentificación de tus usuarios en cuestión de minutos.

Realtime Database: Ya no necesitas crear un servidor con las bases de datos y acceder a él mediante servicios Web. Firebase nos ofrece una solución muy sencilla, fundada en bases de datos NoSQL, en las que todo se almacena en una estructura similar a JSON. Además, son bases de datos en tiempo real. Cualquier cambio en un nodo de la base de datos puede ser automáticamente sincronizado a nuestra aplicación.

Si un usuario pierde la conexión a Internet, podrá usar la caché local para publicar y almacenar cambios. Cuando el dispositivo se vuelve a conectar, los datos de la caché son almacenados.

Cloud Firestore: Una alternativa a Realtime Database, todavía en fase beta. Es muy parecida, sigue siendo NoSQL, de tiempo real y con un API parecido, aunque incorpora un nuevo modelo de datos más intuitivo. Cloud Firestore también cuenta con consultas más ricas y rápidas y resulta más escalable. Como único inconveniente encontramos una mayor latencia.

Cloud Functions: Puedes ejecutar código en el lado de servidor cuando se produzcan ciertos eventos. Por ejemplo, un cambio en la base de da-

tos, la autentificación de un usuario o un evento en Analytics. Este código ha de ser escrito en forma de función de JavaScript y será ejecutado en un entorno de Node.*js* seguro y administrado. Desde Cloud Functions tenemos acceso a todos los servicios de Firebase: bases de datos, notificaciones…

Cloud Storage: Almacena ficheros en la nube con el máximo nivel de disponibilidad. Las transferencias de ficheros pueden ser detenidas o reanudadas según la conectividad del dispositivo. Aprovecha las ventajas de la autentificación de usuario para dar los permisos adecuados a cada fichero. Se realiza redundancia multirregional, lo que garantiza una mínima latencia, aunque se dispare el número de acceso incluso en regiones muy distantes.

Hosting: Crea tu sitio o aplicación Web de forma sencilla. No importa dónde estén tus usuarios, gracias al almacenamiento con redundancia multirregional (CDN) la descarga será rápida en todo el mundo. Utiliza certificados SSL gratuitos para tus propios dominios.

ML Kit: Saca provecho de la experiencia de Google en aprendizaje automático para aplicarlo en la resolución de los problemas planteados en tu aplicación. Si no tienes experiencia en este campo, puedes implementar la funcionalidad que necesitas con solo unas pocas líneas de código. Si eres expertos en redes neuronales u optimización de modelos utiliza TensorFlow Lite para crear tus propias soluciones en tu aplicación móvil.

1.2.1.2. Herramientas de comprobación

Crashlytics: Descubre problemas de tu app para distintos dispositivos y así poderlos corregir lo antes posible. Resulta muy fácil de integrar, tanto en iOS como en Android. Desde el panel de control, podrás ver estos errores agrupados y por orden de importancia. Soluciona los problemas a partir de los potentes informes de fallos en tiempo real.

Crash Reporting: Ha sido reemplazado por Crashlytics.

Test Lab para Android: Para asegurarte de que tu aplicación funciona correctamente, gracias a este servicio, podrás probarla tanto en dispositivos

reales como virtuales. Puedes usarlo desde Android Studio, la consola Web o un entorno de integración continua. Aunque no hayas escrito pruebas de comprobación, puedes usar el rastreador, Robo, para que navegue por tu app de forma automática y detecte errores potenciales.

Performance Monitoring: Descubre si tus usuarios perciben que la aplicación funciona de forma fluida. Obtén estadísticas de rendimiento, tanto personalizadas, como automáticas. Conoce cómo el acceso a Internet afecta a los usuarios. Desde la consola podrás realizar el análisis de los resultados y detectar el origen de los problemas.

1.2.1.3. Herramientas de análisis e interacción con los usuarios

Google Analytics: Analiza toda la información sobre tu app (quién la instala, cómo la utiliza, qué notificaciones ha pulsado…) para así poder tomar las mejores decisiones. Descubre aspectos clave como quién, dónde o cuándo, para así poder averiguar el porqué. Analytics integra información del resto de herramientas como Invites, AdWords, AdMob… Se capturan automáticamente ciertos eventos sobre el uso de tu app, pero también puedes crear tus propios eventos personalizados.

Predictions: Utilizando herramientas de aprendizaje automático, Firebase crea automáticamente grupos de usuarios según las predicciones de su comportamiento. Con estas predicciones puedes mejorar la experiencia de usuario, aumentar los ingresos o la retención. Existen predicciones predeterminadas, como “harán compras” o “se mantendrán en la app”, pero también puedes crear tus propias predicciones personalizadas, como “pasará el primer nivel antes de una semana”.

Cloud Messaging: Envía mensajes entre aplicaciones (FCM) de forma gratuita y entre diferentes plataformas. También puedes enviar notificaciones a tus usuarios directamente desde la consola. Estas pueden contener datos personalizados o una hora de activación adaptada al usuario. Además, puedes analizar los resultados de conversiones desde Analytics.

Dynamic Links: Cuando te pasan un enlace a un vídeo de YouTube, la forma habitual de abrirlo es a través del navegador Web. Sin embargo, es posible que tengas instalada la app de YouTube, que te aportaría una mejor experiencia de usuario. Gracias a los Dynamics Links vas a poder definir URL asociados a

tu aplicación, de forma que el usuario pueda abrir el contenido desde esta. En caso de no tenerla instalada, se le puede sugerir que la instale, o si no está interesado o estás en un sistema no Android, abrir el contenido desde la Web.

Invites: Aumenta la visibilidad de tu aplicación haciendo que tus usuarios envíen invitaciones a sus amigos. Puedes promover este comportamiento con algún tipo de regalo. Las invitaciones pueden ser enviadas por correo electrónico o SMS. Al basarse en Dynamic Links, cuando un usuario reciba la invitación, si tiene instalada la app, se abrirá la actividad correspondiente; en caso contrario, se solicitará la instalación.

Remote Config: Cambia el comportamiento y aspecto de tu app desde la consola, sin tener que lanzar una nueva actualización. Puedes lanzar nuevo contenido, personalizar por segmentos de usuarios o realizar pruebas para validar mejoras.

App Indexing: Consigue que la búsqueda de Google muestre enlaces a tu app cuando los usuarios buscan contenido de tu app. Además, se mejorará el posicionamiento conseguido con tu app.

AdMob: Añade anuncios en tus apps para conseguir ingresos. La plataforma de publicidad móvil de Google ahora está integrada dentro de Firebase.

AdWords: Crea campañas de anuncios en el buscador de Google para dar a conocer tu aplicación y aumentar el número de instalaciones.

1.2.2. Agregar Firebase en un proyecto

En este curso vamos a continuar trabajando con el proyecto “Mis Lugares”, el cual solo permitía almacenar la información en local (en una lista en memoria o en una base de datos SQLite). A lo largo del curso vamos a hacer que esta aplicación trabaje con una base de datos Firebase almacenada en la nube. Además, integraremos muchos de los servicios disponibles en Firebase.

El primer paso va a ser crear un nuevo proyecto en la consola de Firebase y añadir en la aplicación todas las librerías necesarias.

Existen dos formas de agregar un proyecto a Firebase, manual y automática. Para aprender correctamente dónde se encuentra la información relevante reco-

mendamos realizar la manual. Una vez que controlemos el proceso, la automática resulta más rápida.

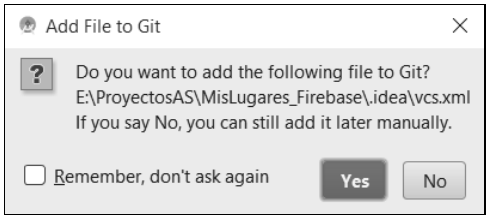
Ejercicio: *Añadir Firebase a "Mis Lugares" de forma manual*

1. Si has realizado el curso de EdX "Android: Introducción a la Programación", puedes utilizar el proyecto que creaste. En caso contrario, o si realizaste muchos cambios, desde Android Studio accede a *File / New / Project from Version Control / GitHub*. Indica el siguiente URL:

   https://github.com/jesus-tomas-girones/MisLugares_base.git

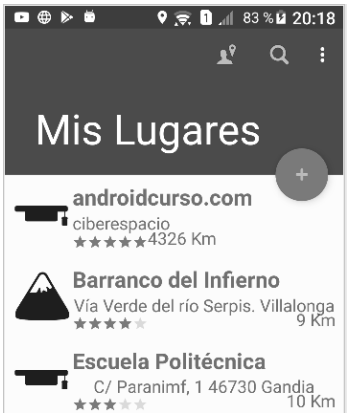
   Puedes crear el proyecto en el directorio que prefieras. Pulsa en **Clone**.

2. Aparecerá el siguiente cuadro de dialogo:



   Cada vez que crees un nuevo fichero en el proyecto, te preguntará si quieres añadirlo al repositorio Git. No tiene sentido que escribas en este Git, por lo que has de indicar **No**. Puede ser una buena idea recordar esta acción, para que no te vuelva a preguntar.

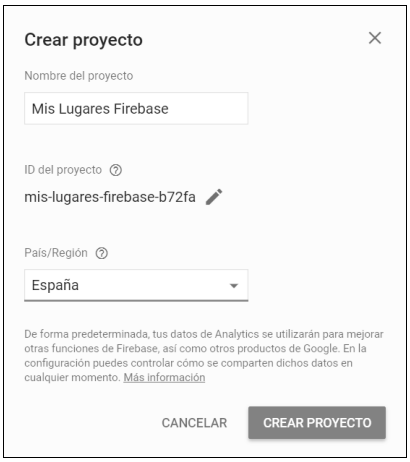
3. Ejecuta el proyecto y verifica que funciona correctamente:



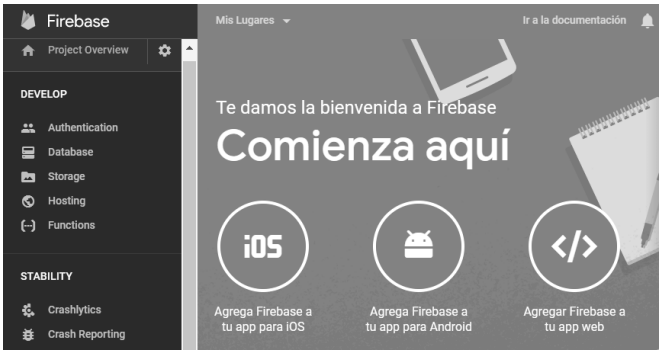
4. Accede a la consola de Firebase (https://console.firebase.google.com). Necesitarás una cuenta de Google.

5. Pulsa en **Agregar proyecto**. Un proyecto puede dar soporte a varias aplicaciones en distintas plataformas como Android, iOS y Web. Toda aplicación que acceda a Firebase ha de registrarse en un proyecto.

6. Introduce como nombre de proyecto, “Mis Lugares”, y un país de referencia. Esta última información se usa principalmente para determinar la moneda en la que se mostrarán los ingresos:

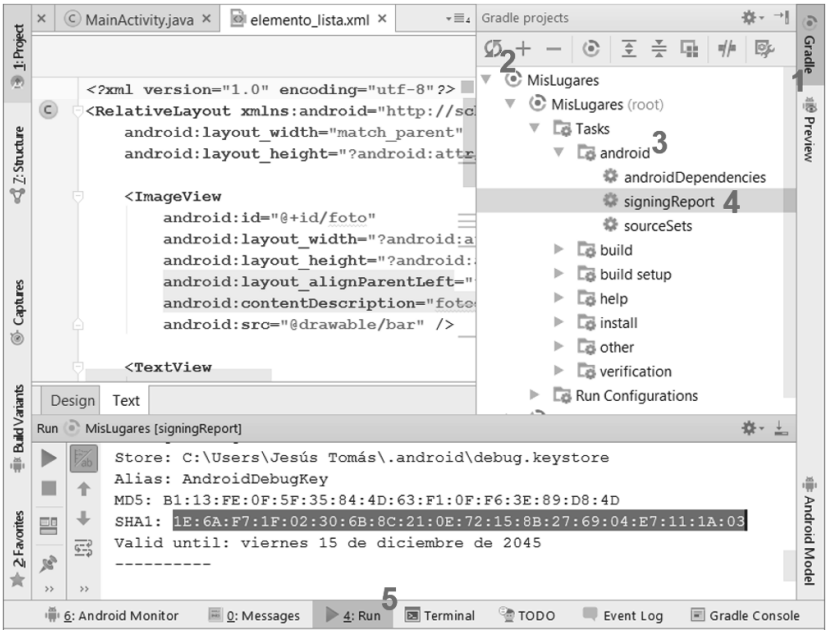


7. Accederás a la consola del proyecto:



8. Pulsa el botón **Agrega Firebase a tu aplicación de Android**.

Añade el nombre del paquete de la aplicación. El apodo, o *nickname*, es opcional. Permite tener varias aplicaciones con un mismo nombre de paquete. En nuestro proyecto no va a ser necesario. La firma SHA1, también es opcional. La necesitaremos para trabajar con autentificación con Google y teléfono, links dinámicos e Invites. Para obtenerla puedes seguir las instrucciones que se muestran al pulsar en el icono de interrogación. Otra alternativa es obtener esta firma directamente desde Android Studio. Si prefieres esta alternativa, sigue los siguientes 5 pasos:

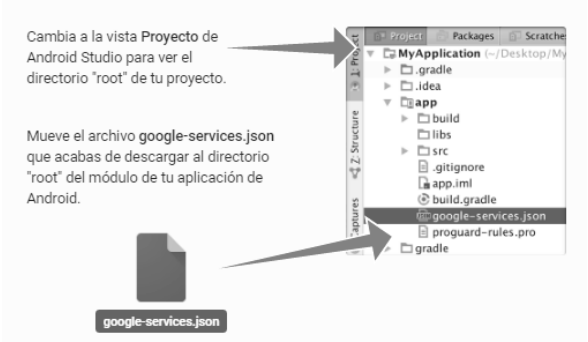


9. Desde Android Studio, pulsa en el botón **Gradle** del panel de la derecha.
10. Una vez abierta la ventana GRADLE, pulsa el botón **Refresh**.
11. En el desplegable abre la ruta *<Nombre del proyecto> / Task / android*.
12. Haz doble clic en **SigningReport**.
13. En la ventana RUN aparecerá la firma digital SHA1 (si es necesario, pulsa el botón). Cópiala al portapapeles, pégala en la consola de Firebase y pulsa en **Registrar app**.

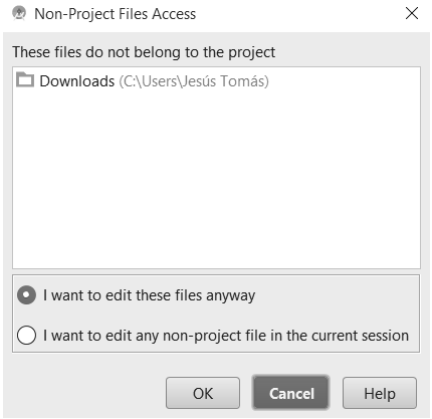
    NOTA: *La firma que has pegado es de debug. Cuando firmes la aplicación para su publicación con la firma definitiva, tendrás que volver a la consola de Firebase y cambiar esta firma.*

    NOTA: *La firma de debug es diferente en cada ordenador donde hayas instalado Android Studio. Si cambias de ordenador, tendrás que volver a la consola de Firebase y cambiar esta firma.*

14. El segundo paso te permite descargar el fichero *google-services.json*. Consiste en un archivo de configuración que hemos de copiar en la carpeta del módulo de nuestra aplicación. Este módulo suele llamarse *app*.



Si aparece una ventana similar a la siguiente pulsa en **OK**.



15. El tercer y último paso consiste en añadir algunas configuraciones en el Gradle del proyecto:

El complemento de los servicios de Google para Gradle carga el archivo `google-services.json` que acabas de descargar. Para poder usar el complemento, debes modificar los archivos build.gradle.

1. **build.gradle de proyecto** (`<project>/build.gradle`):

```
buildscript {
  dependencies {
    // Add this line
    classpath 'com.google.gms:google-services:3.1.0'
  }
}
```

2. **build.gradle de aplicación** (`<project>/<app-module>/build.gradle`):

```
...
// Add to the bottom of the file
apply plugin: 'com.google.gms.google-services'
```

incluye Analytics en la configuración predeterminada

3. Por último, presiona **Sincronizar ahora** en la barra que aparece en el entorno IDE:

Gradle files have changed sir | Sync now

16. En *build.gradle (Project:Audiolibros)* añade:

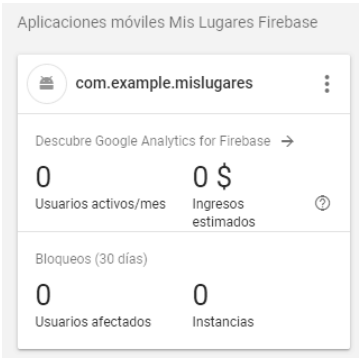
```
buildscript {
   …
   dependencies {
      …
      classpath 'com.google.gms:google-services:3.1.0'
   }
}
…
```

17. En *build.gradle (Module:app)* añade al final:

```
apply plugin: 'com.google.gms.google-services'
```

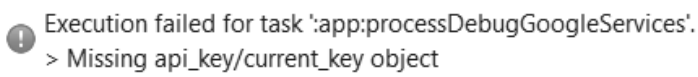
18. Cuando cambias los ficheros de Gradle has de sincronizar el proyecto. Te aparecerá un mensaje similar al siguiente. Pulsa en **Sync Now**.

19. En la consola de Firebase pulsa en el botón **Finalizar**. Aparecerá la siguiente información:



20. Ejecuta el proyecto para verificar que todo funciona correctamente.

NOTA: *Si te aparece el siguiente error:*

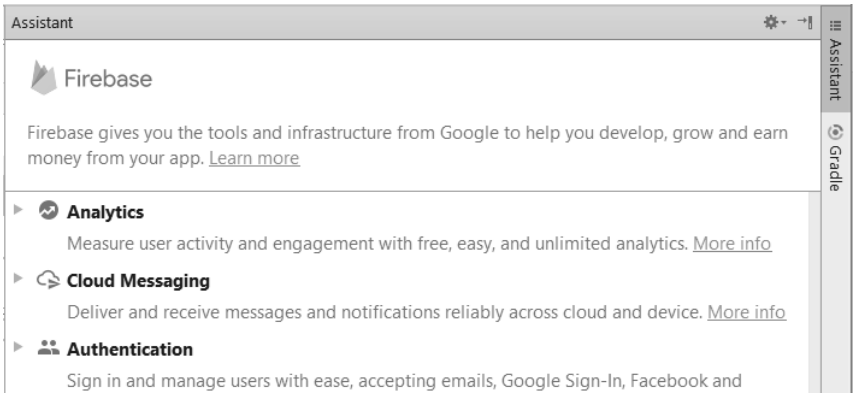


descarga de nuevo el fichero google-services.json. Para descargarlo pulsa en los tres puntos que aparecen junto a com.example.audiolibros de la captura anterior, selecciona ***Configuración*** *y busca el nombre del fichero.*

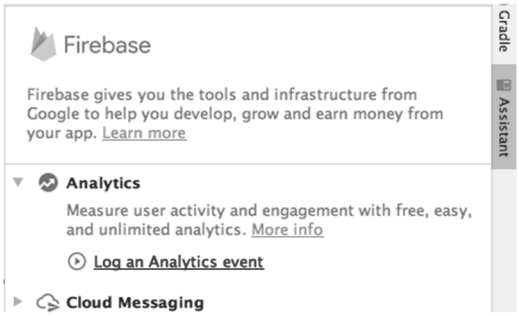
Ejercicio: *Añadir Firebase a un proyecto a de forma automática*

1. Accede a la consola de Firebase (https://console.firebase.google.com) y crea una cuenta, si no tienes una creada.

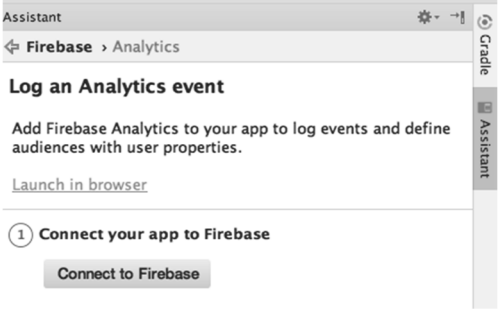
2. Abre o crea el proyecto al que quieras añadir Firebase.
3. Selecciona el menú de *Tools / Firebase*. Aparecerá un asistente que te permitirá integrar rápidamente la mayoría de servicios de Firebase:



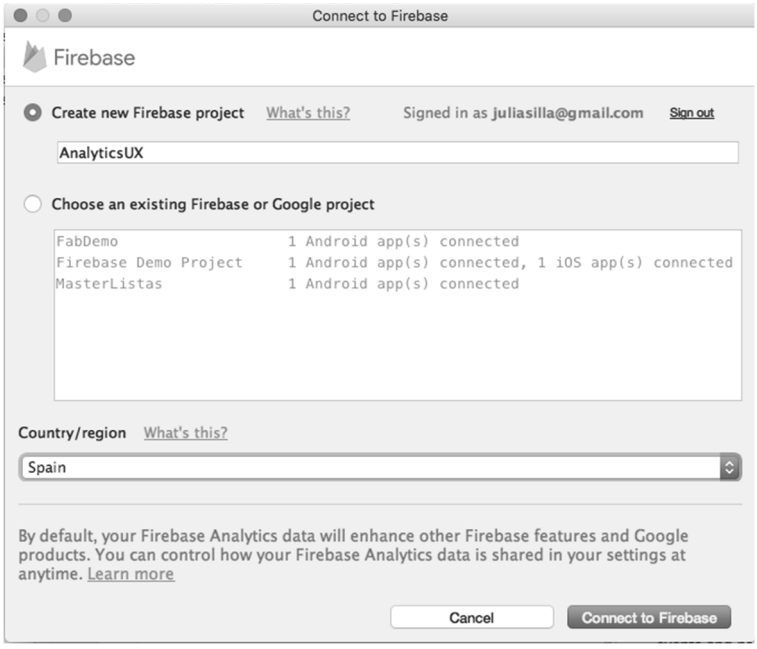
4. A modo de ejemplo probaremos el primero, Analytics. Selecciona *Analytics* / *Log an Analytics event*:



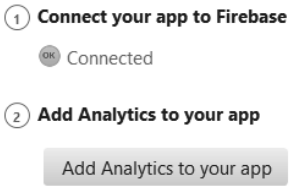
5. Verás que aparecen una serie de pasos que vamos a seguir. Pulsa el botón **Connect to Firebase** e indica tus credenciales:



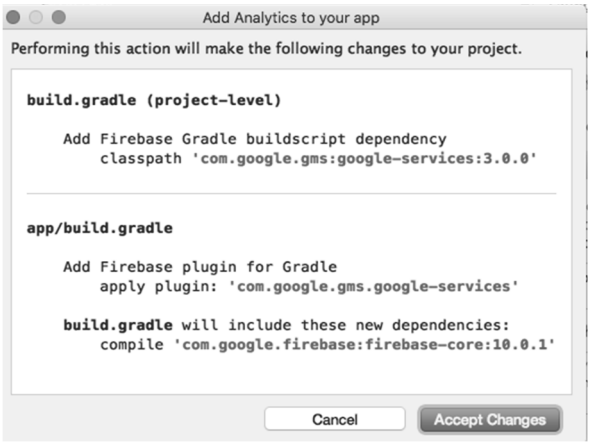
6. Se creará automáticamente un proyecto en Firebase. Acepta el nombre de la aplicación y la región.



7. Pasamos al segundo punto. Pulsa en **Add Analytics to your app**:

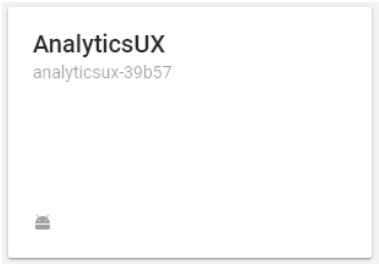


8. Acepta los cambios para añadir las dependencias automáticamente:



9. El siguiente paso de la lista es para generar eventos Log. De momento puedes saltarlo, ya que lo explicaremos más adelante.

10. Si accedes a la consola de Firebase, nuestro proyecto aparecerá en breve:



1.3. Autentificación con Firebase

Realizar un proceso de autentificación a través de Internet es extremadamente costoso. La trasferencia ha de realizarse de manera encriptada a través de https. El almacenamiento de las credenciales ha de cumplir todas las medidas de seguridad. Se ha de implementar un sistema de caché por si los usuarios pierden el acceso a Internet. Se necesita un backend, para gestionar los usuarios, envío de correos y la restauración de contraseñas. Además, resulta interesante implementar pasarelas para delegar la autentificación usando usuarios de redes sociales. Con Firebase no es necesario enfrentarse a todos estos problemas, ya que nos proporciona una solución integrada y totalmente automática.

Otro aspecto a destacar es que la responsabilidad de almacenar las credenciales es de Google. Por lo tanto, no hemos de preocuparnos de posibles fallos de seguridad en la custodia de estas credenciales.

Finalmente, hay que indicar que disponemos de gran variedad de sistemas de autentificación. Además de la autentificación con correo y contraseña, podemos autentificar utilizando el número de teléfono del usuario, realizar una autentificación personalizada, conectándonos a un sistema de autentificación propio, o usuario anónimo, muy interesante si el usuario no quiere darnos sus credenciales hasta estar seguro que va a utilizar la aplicación.

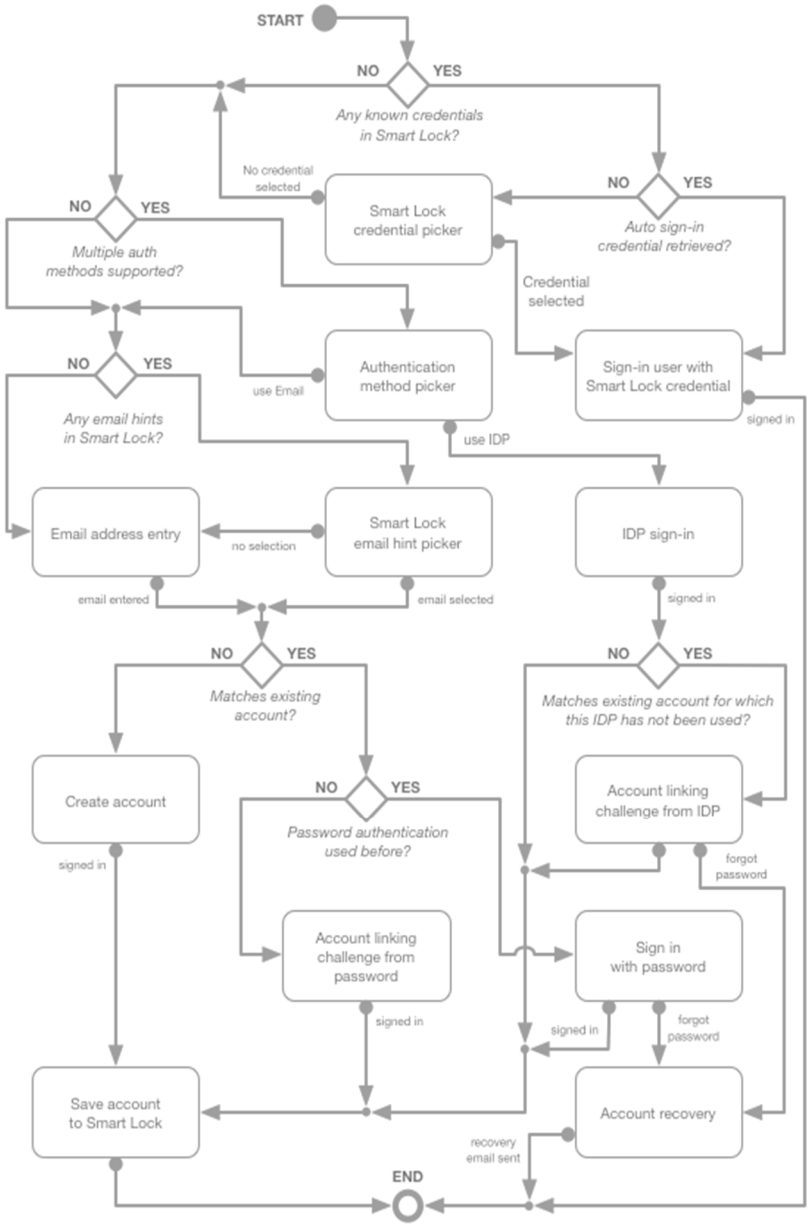
No obstante, la alternativa preferida por muchos usuarios es utilizar el mismo usuario que tienen en redes sociales, y así no tener que memorizar más contraseñas. Vamos a tener soporte para Google, Facebook, Twitter y GitHub.

1.3.1. Autentificación con FirebaseUI

FirebaseUI complementa al SDK, es una biblioteca de código abierto que incluye elementos de interfaz gráfica que podemos usar para realizar acciones frecuentes con autentificación, bases de datos o almacenamiento.

Resulta especialmente útil para autentificación. La librería se encarga de todo, incluso permite al usuario escoger el proveedor de autentificación. No obstante, cuando se presenten problemas va a resultar interesante saber cómo se realiza el proceso de autentificación. También cuando queramos una autentificación personalizada a nuestras necesidades.

Este proceso realizado por FirebaseUI se explica en el siguiente diagrama:

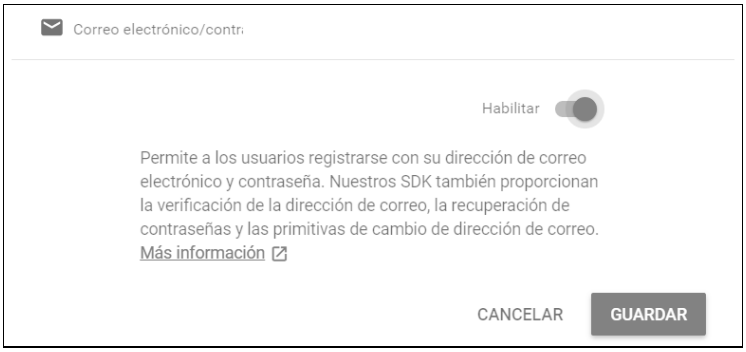


1.3.1.1. Autentificación por correo y Google

Vamos a realizar dos métodos de identificación en un mismo ejercicio, dado que el trabajo a realizar es similar. Si deseas solo uno de estos métodos de identificación, puedes anular el que no te interese.

Ejercicio: Autentificación por correo y Google

1. Accede a la consola de Firebase y en el proyecto *Mis Lugares / Authentication / Método de acceso*, selecciona *Correo electrónico/contraseña*, pulsa en **Habilitar** y luego en **Guardar**:



2. Habilita también como método de acceso las cuentas de Google. No es necesario que actives ninguna opción avanzada. Recuerda que es necesario haber introducido la firma SHA1 en los datos de la aplicación.

3. Añade en *build.gradle (Project)* la línea subrayada:

```
allprojects {
    repositories {
        jcenter()
        maven { url 'https://maven.google.com' }
    }
}
```

FirebaseUI necesita tener acceso a la librería Fabric al ser utilizada por Facebook y Twitter.

4. En *build.gradle (Module: app)* asegúrate de que la versión mínima es 16 o más e incluye la siguiente dependencia:

```
android {
   defaultConfig {
      minSdkVersion 16
      …
dependencies {
   implementation 'com.firebaseui:firebase-ui-auth:3.1.3'
}
```

5. Añade en *AndroidManifest.xml* las líneas subrayadas:

```
<manifest …>
   <application
      android:supportsRtl="true" …>
      …
```

```
        <activity android:name=".LoginActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        …
```

Para que la librería que acabamos de añadir funcione correctamente, tenemos que eliminar el atributo `suportRtl`. Además, registramos una nueva actividad (`LoginActivity`) que será lanzada al iniciar la aplicación.

6. Elimina el `<inter-filter>` de la actividad `MainActivity` que anteriormente se lanzaba al iniciar la aplicación.

   NOTA: *En caso de tener una actividad de splash, es posible que quieras lanzar la actividad para el Login tras esta. En tal caso, modifica estos pasos para adaptarlos a tu aplicación.*

7. Crea la clase `LoginActivity` con el siguiente código:

```
public class LoginActivity extends AppCompatActivity {
    private static final int RC_SIGN_IN = 123;

    @Override protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        login();
    }

    private void login() {
        FirebaseUser usuario = FirebaseAuth.getInstance().getCurrentUser();
        if (usuario != null) {
            Toast.makeText(this, "inicia sesión: " +
                usuario.getDisplayName()+" - "+ usuario.getEmail()+" - "+
                usuario.getProviders().get(0),Toast.LENGTH_LONG).show();
            Intent i = new Intent(this, MainActivity.class);
            i.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP
                      | Intent.FLAG_ACTIVITY_NEW_TASK
                      | Intent.FLAG_ACTIVITY_CLEAR_TASK);
            startActivity(i);
        } else {
            startActivityForResult(AuthUI.getInstance()
             .createSignInIntentBuilder()
             .setAvailableProviders(Arrays.asList(
               new AuthUI.IdpConfig.Builder(AuthUI.EMAIL_PROVIDER).build(),
               new AuthUI.IdpConfig.Builder(AuthUI.GOOGLE_PROVIDER).build()))
             .setIsSmartLockEnabled(false)
             .build(), RC_SIGN_IN);
          }
     }
```

Lo primero que llama la atención de esta actividad es que no se llama al método `setContentView()`. La vista asociada a la actividad se creará solo si es necesario, además va a ser FirebaseUI quien cree la vista asociada a la

actividad. En el método `login()` verificamos si Firebase ya tiene un usuario validado utilizando `getCurrentUser()`. En caso afirmativo, mostramos en un `Toast` los datos del usuario y arrancamos la actividad `MainActivity`.

En caso de no existir el usuario, vas a lanzar una actividad para que el usuario seleccione el método de autentificación. El método `createSignInIntentBuilder()` va a crear la intención adecuada. Podemos configurarla con los proveedores que nos interese. En el ejemplo, correo electrónico y Google.

8. Añade el siguiente método:

```
@Override
public void onActivityResult(int requestCode, int resultCode,Intent data){
   super.onActivityResult(requestCode, resultCode, data);
   if (requestCode == RC_SIGN_IN) {
      if (resultCode == ResultCodes.OK) {
         login();
         finish();
      } else {
         IdpResponse response = IdpResponse.fromResultIntent(data);
         if (response == null) {
            Toast.makeText(this,"Cancelado",Toast.LENGTH_LONG).show();
            return;
         } else if (response.getErrorCode() == ErrorCodes.NO_NETWORK) {
            Toast.makeText(this,"Sin conexión a Internet",
                                                Toast.LENGTH_LONG).show();
            return;
         } else if (response.getErrorCode() == ErrorCodes.UNKNOWN_ERROR) {
            Toast.makeText(this,"Error desconocido",
                                                Toast.LENGTH_LONG).show();
            return;
         }
      }
   }
}
```

Este método será llamado.

9. Ejecuta la aplicación. El resultado ha de ser similar al siguiente:

10. Tras acceder con varios usuarios, entra en la consola de Firebase. En la pestaña **Usuarios** encontrarás la siguiente información:

11. Cuando en un mismo dispositivo entres por segunda vez ya no te pedirá las credenciales de acceso. Como todavía no hemos implementado la opción de logout, si quieres hacer más pruebas, ves a la consola de Firebase y elimina el usuario creado.

12. Cambia en el método `doLogin()` el valor de `setIsSmartLockEnabled(false)` a `true`. Verifica que el sistema recordará los usuarios que han accedido a Mis Lugares y sus contraseñas. Esto permite un cambio de usuario mucho más rápido.

Práctica: Verificación de correo

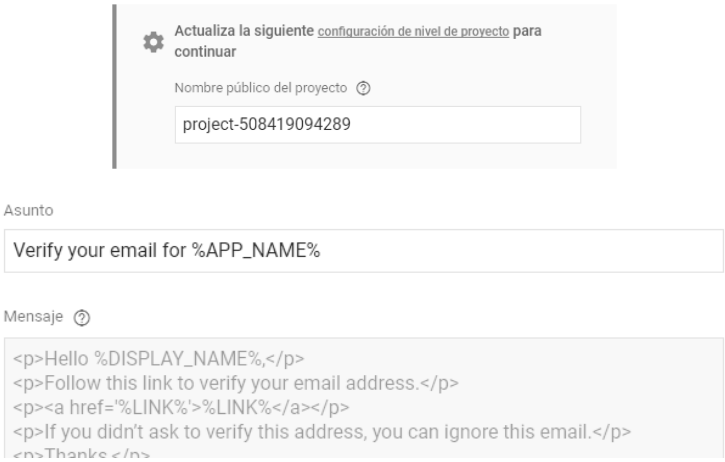
Como has podido observar, si creas una cuenta con correo y contraseña, no es necesario verificar si es válida. Esto significa que algunas personas podrán usar una dirección de correo que no sea suya o que ni siquiera exista. Para evitar esto, Firebase pone a nuestra disposición el envío de correos de confirmación. Esta técnica consiste en el envío de un correo a la dirección que indicó el usuario. El usuario ha de abrir este correo y pulsar en un link para validar su cuenta.

Mediante el método `FirebaseUser.isEmailVerified()` podemos averiguar si ha verificado su dirección de correo. Para que se le envíe un correo de verificación, tenemos el método `FirebaseUser.sendEmailVerification()`.

Con estos dos métodos, haz que la aplicación no deje entrar a los usuarios si no han verificado su cuenta de correo. Para ello, verifica en `login()`, antes de lanzar la actividad principal si el usuario ya está verificado, y en caso negativo envíale un correo de verificación y avísalo de que ha de abrir el correo. Realiza

estas acciones solo si el proveedor de autentificación es correo. Para el resto de proveedores, estas acciones no tienen sentido.

Accede a la consola de Firebase y selecciona *Authentication / Plantillas / Verificación de dirección de correo electrónico*. Modifica el nombre público del proyecto para que sea fácilmente identificable. Modifica el asunto de la plantilla para que el texto esté en castellano. Si tratas de modificar el cuerpo del mensaje, comprobarás que no te lo permite. Aparecerá el siguiente error: "Para evitar el envío de *spam*, no se puede editar el mensaje en esta plantilla de correo".



1.3.1.2. Obtener datos del usuario y cerrar sesión

Obtener los datos del perfil resulta muy sencillo:

```
FirebaseUser usuario = FirebaseAuth.getInstance().getCurrentUser();

String nombre = usuario.getDisplayName();
String correo = usuario.getEmail();
String telefono = usuario.getPhoneNumber();
Uri urlFoto = usuario.getPhotoUrl()
String uid = usuario.getUid();
String proveedor = usuario.getProviderId();
```

No siempre se dispone de toda la información, esto depende del proveedor de autentificación.

NOTA: *Los métodos para acceder o modificar el perfil de usuario y cerrar sesión pertenecen al SDK de Firebase. Se han añadido dentro del apartado de FirebaseUI para secuenciar mejor el aprendizaje.*

Ejercicio: Visualizar los datos del usuario

1. Añade el siguiente ítem de menú en *res / menu / menu_main.xml*:

```
<item android:id="@+id/menu_usuario"
    android:title="Perfil de Usuario"
    android:icon="@android:drawable/ic_menu_info_details"
    android:orderInCategory="1"
    app:showAsAction="always|withText"/>
```

2. En `MainActivity` añade dentro de `onOptionsItemSelected()` el siguiente código:

```
if (id == R.id.menu_usuario) {
   Intent intent = new Intent(this, UsuarioActivity.class);
   startActivity(intent);
}
```

3. Crea `UsuarioActivity` con el siguiente código:

```
public class UsuarioActivity  extends AppCompatActivity {
   @Override protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_usuario);
   }
}
```

4. Crea *res/layout/activity_usuario.xml* con el siguiente código:

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal" >
    <fragment
        android:id="@+id/usuario_fragment"
        android:name="com.example.mislugares.UsuarioFragment"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:clickable="true"/>
</LinearLayout>
```

5. Crea `UsuarioFragment` con el siguiente código:

```
public class UsuarioFragment extends Fragment {
   @Override public View onCreateView(LayoutInflater inflador,
                        ViewGroup contenedor, Bundle savedInstanceState) {
      View vista = inflador.inflate(R.layout.fragment_usuario,
                         contenedor, false);
      FirebaseUser usuario = FirebaseAuth.getInstance().getCurrentUser();
      TextView nombre = (TextView) vista.findViewById(R.id.nombre);
      nombre.setText(usuario.getDisplayName());
      return vista;
   }
}
```

6. Crea *res/layout/fragment_usuario.xml* con el siguiente código:

```
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/scrollView1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical" >
        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="horizontal" >
            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Nombre de usuario: " />
            <TextView
                android:id="@+id/nombre"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content" />
        </LinearLayout>
    </LinearLayout>
</ScrollView>
```

7. Registra la actividad que acabas de crear en *AndroidManifest.xml*:

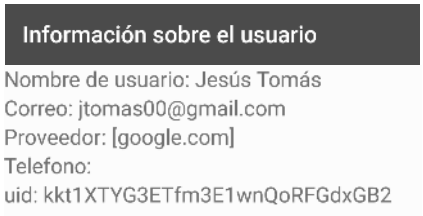
```
<activity android:name=".UsuarioActivity"
          android:label="Información sobre el usuario" />
```

8. Verifica el resultado.

9. Visualiza en la actividad anterior otros datos del usuario utilizando los métodos `getEmail()`, `getProviders()`, `getPhoneNumber()` y `getUid()`. El método `getProviders()` ofrece una lista de proveedores, para visualizarlos, utiliza `toString()`:

```
TextView proveedores = (TextView) vista.findViewById(R.id.proveedores);
proveedores.setText(usuario.getProviders().toString());
```

El resultado ha de ser similar a:



Ejercicio: Cerrar sesión

1. Añade un botón para cerrar sesión en *fragment_usuario.xml*:

```
<Button
    android:id="@+id/btn_cerrar_sesion"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Cerrar sesión"
    android:padding="8dp"
    android:layout_marginRight="16dp"
    android:layout_marginTop="16dp"
    android:background="#d0021b"
    android:textColor="#fff"/>
```

2. En `UsuarioFragment`, añade en `onCreateView()` el siguiente código:

```
Button cerrarSesion =(Button) vista.findViewById(R.id.btn_cerrar_sesion);
cerrarSesion.setOnClickListener(new View.OnClickListener() {
   public void onClick(View view) {
      AuthUI.getInstance().signOut(getActivity())
         .addOnCompleteListener(new OnCompleteListener<Void>() {
            @Override
            public void onComplete(@NonNull Task<Void> task) {
               Intent i = new Intent(getActivity(),LoginActivity.class);
               i.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP
                           | Intent.FLAG_ACTIVITY_NEW_TASK
                           | Intent.FLAG_ACTIVITY_CLEAR_TASK);
               startActivity(i);
               getActivity().finish();
            }
         });
   }
});
```

3. Verifica el resultado.

Ejercicio: Añadir fotografía de usuario

Las cuentas de Google, Facebook y Twitter suelen tener una fotografía de usuario asociada. En este ejercicio vamos a ver cómo podemos acceder a esta fotografía.

1. Añade la siguiente dependencia en *build.gradle (Module: app)*:

```
compile 'com.android.volley:volley:1.0.0'
```

2. Añade en *fragment_usuario.xml*, al principio del `LinearLayout`, el código:

```
<com.android.volley.toolbox.NetworkImageView
    android:id="@+id/imagen"
    android:layout_width="120dp"
    android:layout_height="120dp"
    android:paddingTop="0dp"
    android:src="@android:drawable/sym_def_app_icon" />
```

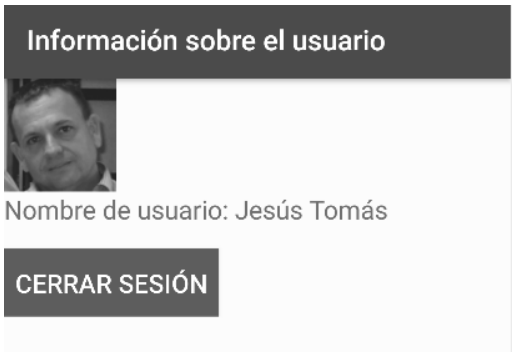
Vamos a cargar la imagen a un tamaño fijo. Además, cargaremos la imagen de Internet usando un `NetworkImageView`.

3. En `UsuarioFragment`, añade en `onCreateView()`,el siguiente código:

```
// Inicialización Volley  (Hacer solo una vez en Singleton o Applicaction)
RequestQueue colaPeticiones = Volley.newRequestQueue(getActivity()
                                                    .getApplicationContext());
ImageLoader lectorImagenes = new ImageLoader(colaPeticiones,
        new ImageLoader.ImageCache() {
            private final LruCache<String, Bitmap> cache =
                    new LruCache<String, Bitmap>(10);
            public void putBitmap(String url, Bitmap bitmap) {
                cache.put(url, bitmap);
            }
            public Bitmap getBitmap(String url) {
                return cache.get(url);
            }
        });

// Foto de usuario
Uri urlImagen = usuario.getPhotoUrl();
if (urlImagen != null) {
    NetworkImageView fotoUsuario = (NetworkImageView)
            vista.findViewById(R.id.imagen);
    fotoUsuario.setImageUrl(urlImagen.toString(), lectorImagenes);
}
```

4. Verifica que el resultado es similar al siguiente:



1.3.1.3. Métodos para cambiar el perfil de usuario

Cambiar la información de perfil de un usuario resulta sencillo. Mira este ejemplo:

```
FirebaseUser usuario = FirebaseAuth.getInstance().getCurrentUser();
UserProfileChangeRequest perfil = new UserProfileChangeRequest.Builder()
     .setDisplayName("Nuevo nombre usuario")
     .setPhotoUri(Uri.parse("https://www.ejemplo.com/usuario/foto.jpg"))
     .build();
usuario.updateProfile(perfil);
usuario.updateEmail("nuevo.correo@ejemplo.com");
```

```
usuario.updatePassword("nueva cotraseña")
        .addOnCompleteListener(new OnCompleteListener<Void>() {
           @Override
           public void onComplete(@NonNull Task<Void> task) {
              if (!task.isSuccessful()) {
                 Log.e("MisLugares", "Acción incorrecta");
              }
           }
        });
```

Los tres métodos `update` pueden añadir un escuchador `OnCompleteListener` para asegurarte que ha funcionado correctamente. Para mayor claridad, solo se ha añadido en el último.

Por razones de seguridad cambiar el correo, la contraseña o borrar la cuenta (`.delete()`) no pueden realizarse si el usuario no ha accedido recientemente. En caso contrario, se creará una `FirebaseAuthRecentLoginRequiredException`. Cuando esto suceda, debes volver a autentificar al usuario. Para ello, pide de nuevo al usuario las credenciales de acceso y pásalas a `reauthenticate`:

```
AuthCredential credential = EmailAuthProvider
        .getCredential("viejo.correo@ejemplo.com", "vieja contraseña");
usuario.reauthenticate(credential);
```

En este ejemplo se muestra cómo obtener las credenciales para correo y contraseña. A lo largo del capítulo se mostrará cómo hacerlo para otros proveedores. Es interesante que añadas el escuchador al método `reauthenticate` para informar al usuario si el cambio ha sido satisfactorio.

Desafío: Modificar datos de perfil

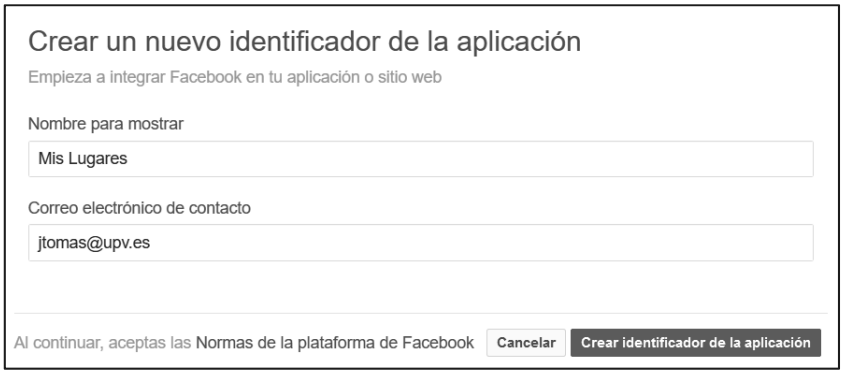
Utilizando estos métodos, modifica `UsuarioFragment` para permitir al usuario cambiar todos o algunos de los datos de su perfil. Puedes utilizar el planteamiento que prefieras. Puedes realizarlo para el proveedor correo y contraseña o también para otros proveedores. Es importante que informes al usuario si el cambio se ha realizado con éxito o incluso, en función del proveedor, si se va a poder realizar el cambio. Es recomendable realizar este desafío al final de la unidad. Se puede utilizar el esquema de verificación de formularios propuesto en el siguiente apartado.

1.3.1.4. Autentificación por Facebook y Twitter

Ejercicio: Autentificación con Facebook

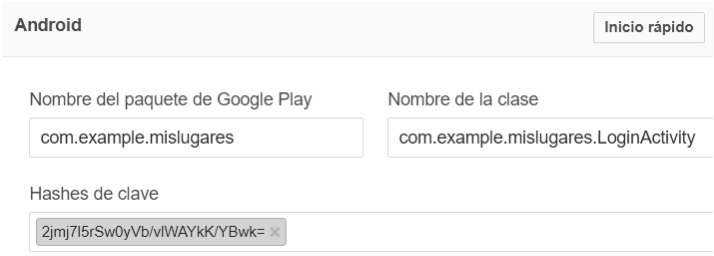
Para poder validar usuarios con Facebook en tu aplicación vas a tener que darla de alta en la consola de Facebook. Has de tener en cuenta que mientras tu aplicación esté en fase de desarrollo, no podrás probarla con cualquier usuario de Facebook. Solo podrás hacerlo con los que hayas dado de alta como evaluadores.

1. Entra en el sitio de Facebook para desarrolladores (https://developers.facebook.com/?locale=es_ES). Pulsa el en el botón **Entrar** e introduce o crea un usuario de Facebook.
2. Si no estás registrado como desarrollador de Facebook, pulsa en el botón **Regístrate**.
3. Pulsa en el botón **Crear aplicación** (si no aparece, usa *Mis aplicaciones / Añadir una nueva aplicación*) y rellena los siguientes datos:



4. Pulsa en **Crear identificador de la aplicación** y sigue las instrucciones.
5. Entrarás en la consola de la aplicación. En el menú de la izquierda, selecciona *Configuración / Básica*:

6. Pulsa en **Agregar plataforma**, selecciona **Android** e introduce los datos:



7. Para obtener la clave Hashes, utiliza en Windows el siguiente comando:

```
keytool -exportcert –alias androiddebugkey -keystore
"%HOMEPATH%\.android\debug.keystore" | openssl sha1 -binary | openssl base64
```